

## **【Claims】**

**【Claim 1】** A design verification apparatus comprising:

5 a verification software and one or more simulators, wherein said verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one or more design objects in said design code or synthesized net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast while obtaining the visibility, thereby providing both fast  
10 simulation speed and high visibility for debugging.

**【Claim 2】** A design verification method, in which one or more design bugs in the design code or synthesized gate-level net-list are identified and collected by a number of  
15 simulation executions with a number of test benches, comprising;

some of said simulation executions is decomposed into the 1<sup>st</sup> simulation run as the front-state simulation and the post-1<sup>st</sup> simulation runs as the back-stage simulation, verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast while obtaining the visibility, thereby providing both fast simulation speed and high visibility for debugging.

**【Claim 3】** The design verification apparatus and verification method according to Claim 1, Claim 2, Claim 12, Claim 54, or Claim 55 wherein said minimally necessary information includes to save one or more simulation states into files at some fixed interval or, one or more specific simulation times.

**【Claim 4】** The design verification apparatus and verification method according to Claim 1, Claim 2, Claim 12, Claim 54, or Claim 55 wherein the collection of said minimally necessary information at the 1<sup>st</sup> simulation run as the front-state simulation includes to probe on all inputs and inouts in one or more design blocks existed in the design code or synthesized gate-level net-list during the entire simulation and save them into files, and use them as input stimuli at the post-1<sup>st</sup> simulation runs which are the back-stage simulation.

**【Claim 5】** The design verification method according to Claim 3 wherein from said one

or more saved simulation states which have stored at 1<sup>st</sup> simulation run as the front-state simulation, one simulation state is selected and used so that one of post-1<sup>st</sup> simulation runs can start from the simulation state selected and during the simulation all signals and variables in the design or synthesized gat-level net-list are probed.

5

【Claim 6】 The design verification method according to Claim 5 wherein the post-1<sup>st</sup> simulation runs are started with a simulator from two or more saved simulation states one by one sequentially which have stored at 1<sup>st</sup> simulation run as the front-state simulation, one simulation state, which is the latest simulation state in the simulation time, is selected and used so that first of many post-1<sup>st</sup> simulation runs can start from the latest simulation state selected and during the simulation all signals and variables in the design or synthesized gat-level net-list are probed, and more post-1<sup>st</sup> simulation runs are also executed in the reverse order of simulation state saving times.

【Claim 7】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 5, Claim 6, Claim 12, Claim 54, or Claim 55 wherein for the simulation state saving method at one or more simulation times during the 1<sup>st</sup> simulation run, which is the front-stage simulation, the save command or checkpoint command of the simulator is used, and for the method of starting one or more post-1<sup>st</sup> simulation runs which are the back-stage simulation from one of said saved simulation states, the restart command or restore command of the simulator is used.

【Claim 8】 The design verification method according to Claim 4 wherein the collection of said minimally necessary information at the 1<sup>st</sup> simulation run as the front-state simulation includes to probe on all inputs and inouts in one or more design blocks existed in the design code or synthesized gate-level net-list during the entire simulation and save them into files, they are compiled with one or more design blocks existed in the design code or synthesized gate-level net-list in a simulator for generating one or more simulation execution files, and one of them is executed while additional probing is applied to said one or more design blocks existed in the design code or synthesized gate-level net-list at the post-1<sup>st</sup> simulation runs which are the back-stage simulation, thereby providing both fast simulation and visibility inside the design blocks.

【Claim 9】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 5, Claim 6, Claim 12, Claim 54, or Claim 55 wherein each of one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is transformed into two or more partial simulation runs which use two or more simulation states saved during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and said two or more partial simulation runs are executed in parallel by initializing the simulators' each state with each of said saved simulation states and independently running two or more simulators which are

installed on two or more computers connected into a computer network, thereby achieving a fast simulation.

5       【Claim 10】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 5, Claim 6, Claim 12, Claim 54, or Claim 55 wherein each of one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is transformed into two or more partial simulation runs which use two or more executable files of a simulator, which is obtained by compiling two or more design blocks and their dump files for probing values on all inputs of inouts of said two or more design blocks during the entire 1<sup>st</sup> simulation run, which is the front-stage simulation, and said two or more partial simulation runs are executed in parallel by independently running said two or more executable files of a simulator with two or more simulators which are installed on two or more computers connected into a computer network, thereby achieving a fast simulation.

15       【Claim 11】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 5, Claim 6, Claim 12, Claim 54, or Claim 55 wherein each of one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is transformed into two or more partial simulation runs which use two or more executable files of a simulator, which is obtained by compiling two or more design blocks and test benches converted from their dump files for probing values on all inputs of inouts of said two or more design blocks during the entire 1<sup>st</sup> simulation run, which is the front-stage simulation, and said two or more partial simulation runs are executed in parallel by independently running said two or more executable files of a simulator with two or more simulators which are installed on two or more computers connected into a computer network, thereby achieving a fast simulation.

25       【Claim 12】 A design verification method, in which the additional simulation uses the results from previous execution of arbitrary simulation, simulation acceleration, hardware emulation, or prototyping comprising, and said additional simulation is decomposed into the 1<sup>st</sup> simulation run as the front-state simulation and the post-1<sup>st</sup> simulation runs as the back-stage simulation, verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast.

35       【Claim 13】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 4, Claim 5, Claim 9, Claim 10, Claim 11, Claim 12, Claim 23, Claim 24, Claim 25, Claim 27, Claim 54, or Claim 55 wherein the 1<sup>st</sup> simulation run, which is the front-stage

simulation, is a parallel simulation.

5     **【Claim 14】** The design verification method according to Claim 31, or Claim 32 wherein the 1<sup>st</sup> simulation run, which is the front-stage simulation, is executed in cycle-based simulation for entire DUV or one or more design objects in DUV, and one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is executed in event-driven simulation for entire DUV or one or more design objects in DUV.

10     **【Claim 15】** The design verification method according to Claim 31, or Claim 32 wherein the 1<sup>st</sup> simulation run, which is the front-stage simulation, is executed with SystemC simulator for entire DUV or one or more design objects in DUV, and one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is executed with Verilog simulator, VHDL simulator, or SystemVerilog simulator for entire DUV or one or more design objects in DUV.

15     **【Claim 16】** The design verification method according to Claim 31, or Claim 32 wherein the 1<sup>st</sup> simulation run, which is the front-stage simulation, is executed in RTL simulation for entire DUV or one or more design objects in DUV, and one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is executed in gate-level simulation for entire  
20     DUV or one or more design objects in DUV.

25     **【Claim 17】** The design verification method according to Claim 14, Claim 15, or Claim 16 wherein said one or more post-1<sup>st</sup> simulation runs are executed in parallel with two or more simulators.

30     **【Claim 18】** A design verification method, in which one or more design bugs in the design code or synthesized gate-level net-list are identified and corrected by a number of simulation executions with a number of test benches, comprising;  
some of said simulation executions is decomposed into the 1<sup>st</sup> simulation run as the front-state simulation and the post-1<sup>st</sup> simulation runs as the back-stage simulation, verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one  
35     or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast while obtaining the visibility, thereby providing both fast simulation speed and high visibility for debugging.

40     **【Claim 19】** The design verification method according to Claim 14, or Claim 16 wherein

said one or more post-1<sup>st</sup> simulation runs are executed in event-driven simulation at gate-level.

5       **【Claim 20】** The design verification method according to Claim 19 wherein said event-driven simulation at gate-level for said one or more post-1<sup>st</sup> simulation runs is timing simulation by using timing information extracted from placement and routing.

10       **【Claim 21】** The design verification method according to Claim 14, Claim 15, Claim 19, or Claim 20 wherein the state information of one or more design objects in DUV which require visibility is obtained during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is executed by using said state information obtained.

15       **【Claim 22】** The design verification method according to Claim 14, Claim 15, Claim 19, or Claim 20 wherein the state information of one or more design objects in DUV which require visibility is obtained during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and two or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is executed in parallel by using said state information obtained.

20       **【Claim 23】** The design verification method according to Claim 1, Claim 2, Claim 12, Claim 54, or Claim 55 wherein the collection of said minimally necessary information at the 1<sup>st</sup> simulation run as the front-state simulation includes to save the design states of one or more design objects in DUV which are required said visibility at regular interval or one or more simulation times, and to probe on all inputs and inouts of said one or more  
25       design objects in DUV event by event, cycle by cycle, or transaction by transaction and save them into one or more files to use them as input stimuli for the post-1<sup>st</sup> simulation runs, which is the back-stage simulation, or alternatively to save the test bench states at said regular interval or one or more simulation times.

30       **【Claim 24】** The design verification method according to Claim 1, Claim 2, Claim 12, Claim 54, or Claim 55 wherein the collection of said minimally necessary information at the 1<sup>st</sup> simulation run as the front-state simulation includes to save the design states of one or more design objects in DUV which are required said visibility at regular interval or one or more simulation times, and to probe on all inputs and inouts of said one or more  
35       design objects in DUV event by event, cycle by cycle, or transaction by transaction and save them into one or more files to use them as input stimuli for the post-1<sup>st</sup> simulation runs, which is the back-stage simulation.

40       **【Claim 25】** The design verification method according to Claim 23, or Claim 24 wherein from said one or more saved design states of one or more design objects in DUV and, if

necessary, saved test bench states which have stored at one or more simulation times in the 1<sup>st</sup> simulation run as the front-state simulation, one or more state is selected and used so that one or more of the post-1<sup>st</sup> simulation runs can start from the design states and test bench states selected and during the simulation all signals and variables in the design or synthesized gat-level net-list are probed.

5  
10  
15  
【Claim 26】 The design verification method according to Claim 25 wherein the post-1<sup>st</sup> simulation runs are started with a simulator from two or more saved design states and if necessary, test bench states one by one sequentially which have stored at 1<sup>st</sup> simulation run as the front-state simulation, one design state and if necessary test bench state, which is the state saved at the latest simulation time, is selected and used so that first of many post-1<sup>st</sup> simulation runs can start from said state selected and during the simulation all signals and variables in the design or synthesized gat-level net-list are probed, and more post-1<sup>st</sup> simulation runs are also executed in the reverse order of state saving times.

20  
25  
30  
【Claim 27】 The design verification method according to Claim 1, Claim 2, Claim 23, Claim 24, Claim 25, Claim 26, Claim 54, or Claim 55 wherein each of one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is transformed into two or more partial simulation runs which use two or more design states and, if necessary, test bench states saved during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and said two or more partial simulation runs are executed in parallel by initializing the simulators' each state with each of said saved states and independently running two or more simulators which are installed on two or more computers connected into a computer network, thereby achieving a fast simulation.

35  
40  
【Claim 28】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 4, Claim 5, Claim 9, Claim 10, Claim 11, Claim 12, Claim 23, Claim 24, Claim 25, Claim 27, Claim 54, or Claim 55 wherein for faster simulation the translated design code, which has different syntax from the original code but is functionally equivalent to the original, is used for the 1<sup>st</sup> simulation run, which is the front-stage simulation.

【Claim 29】 A design verification method, in which one or more design bugs in the design code or synthesized gate-level net-list are identified and corrected by a number of simulation executions with a number of test benches, comprising;  
the design states of one or more design objects in DUV and, in necessary, one or more test bench states are saved at one or more simulation times,

【Claim 30】 The design verification method according to Claim 21, Claim 22, Claim 23, Claim 24, Claim 25, Claim 26, Claim 27, or Claim 29 wherein the minimal design state information of said one or more design objects in DUV which need visibility is used for

said design state information of them.

5       【Claim 31】 The design verification method according to Claim 1, Claim 2, Claim 3, Claim 4, Claim 5, Claim 9, Claim 10, Claim 11, Claim 12, Claim 23, Claim 24, Claim 25, Claim 27, Claim 54, or Claim 55 wherein in the 1<sup>st</sup> simulation run, which is the front-stage simulation, at least one or more design objects has design code abstracted more than those in the post-1<sup>st</sup> simulation runs, which is the back-stage simulation, said 1<sup>st</sup> simulation is executed at the higher level of abstraction than said post-1<sup>st</sup> simulation, thereby achieving more rapid verification.

10       【Claim 32】 The design verification method according to Claim 31 wherein the state information of one or more design objects in DUV which require visibility is obtained during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is executed in parallel with two or  
15       more simulators or in sequence with a single simulator by using said state information obtained.

20       【Claim 33】 The design verification method according to Claim 23, or Claim 24 wherein from said one or more saved design states of one or more design objects in DUV which have stored at 1<sup>st</sup> simulation run as the front-state simulation, one design state is selected and used so that one of post-1<sup>st</sup> simulation runs can start from the design state selected and during the simulation all signals and variables in the design or synthesized gat-level net-list are probed.

25       【Claim 34】 The design verification method according to Claim 33 wherein the post-1<sup>st</sup> simulation runs are started with a simulator from two or more saved design states of one or more design objects in DUV one by one sequentially which have stored at 1<sup>st</sup> simulation run as the front-state simulation, one design state, which is the state saved at the latest simulation time, is selected and used so that first of many post-1<sup>st</sup> simulation  
30       runs can start from the latest design state selected and during the simulation all signals and variables in the design or synthesized gat-level net-list are probed, and more post-1<sup>st</sup> simulation runs are also executed in the reverse order of design state saving times.

35       【Claim 35】 The design verification method according to Claim 1, Claim 2, Claim 23, Claim 24, Claim 33, Claim 34, Claim 54, or Claim 55 wherein each of one or more post-1<sup>st</sup> simulation runs, which is the back-stage simulation, is transformed into two or more partial simulation runs which use two or more design states of one or more design objects in DUV saved during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and said two or more partial simulation runs are executed in parallel by initializing the  
40       simulators' each state with each of said saved states and independently running two or

more simulators which are installed on two or more computers connected into a computer network, thereby achieving a fast simulation.

5       **【Claim 36】** A design verification method, in which one or more design bugs in the design code or synthesized gate-level net-list are identified and corrected by a number of simulation executions with a number of test benches, comprising;  
some of said simulation executions is decomposed into the 1<sup>st</sup> simulation run as the front-state simulation and the post-1<sup>st</sup> simulation runs as the back-stage simulation, verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or  
10       more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast while obtaining the visibility, thereby  
15       providing both fast simulation speed and high visibility for debugging.

20       **【Claim 37】** The design verification method, in which a simulation at the lower level of abstraction is executed rapidly by using the simulation result at the higher level of abstraction.

25       **【Claim 38】** The design verification method according to Claim 37, Claim 57, or Claim 58 wherein said simulation result at the higher level of abstraction contains the state information of one or more design objects.

30       **【Claim 39】** The design verification method according to Claim 38, Claim 64, or Claim 65 wherein state information of one or more design objects is the complete state information.

35       **【Claim 40】** The design verification method according to Claim 38, Claim 64, or Claim 65 wherein state information of one or more design objects is the minimal state information.

40       **【Claim 41】** The design verification method according to Claim 38, Claim 39, Claim 40, Claim 64, or Claim 65 wherein said fast simulation method at the lower level of abstraction is to use two or more simulation in parallel by using said state information of one or more design objects.

45       **【Claim 42】** The design verification method according to Claim 41, or Claim 67 wherein said parallel simulation is temporally parallelized one.



【Claim 43】 The design verification method according to Claim 41, or Claim 67 wherein said parallel simulation is spatially parallelized one.

5       【Claim 44】 The design verification and correction method according to Claim 37 wherein assuming one of two simulation results at the higher level of abstraction and the lower level of abstraction as a reference, comparing the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction, finding the differences between the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction and making them correct, thereby said  
10       two simulation results compatible.

      【Claim 45】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is cycle-based simulation, and said simulation at the lower level of abstraction is event-driven simulation.  
15

      【Claim 46】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is SystemC simulation, and said simulation at the lower level of abstraction is RTL simulation.

20       【Claim 47】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is RTL simulation, and said simulation at the lower level of abstraction is gate-level simulation.

      【Claim 48】 The design verification and correction method according to Claim 47, wherein said gate-level simulation at the lower level of abstraction is timing simulation using the timing information.  
25

      【Claim 49】 The design verification and correction method according to Claim 44, or Claim 59, to find situations where two simulation results at the lower level of abstraction and the higher level of abstraction, the simulation result at the higher level of abstraction is used at the simulation at the lower level of abstraction.  
30

      【Claim 50】 The design verification and correction method according to Claim 44, or Claim 59, to find situations where two simulation results at the lower level of abstraction and the higher level of abstraction, the simulation result at the higher level of abstraction is used at the parallel simulation at the lower level of abstraction.  
35

      【Claim 51】 The design verification and correction method according to Claim 49, or Claim 50, to find situations where two simulation results at the lower level of abstraction and the higher level of abstraction, the simulation result at the higher level of abstraction,  
40

which is used at the simulation at the lower level of abstraction, contains the state information of one or more design objects.

5       **【Claim 52】** The design verification and correction method according to Claim 51, wherein state information of one or more design objects is the complete state information.

**【Claim 53】** The design verification and correction method according to Claim 51, wherein state information of one or more design objects is the minimal state information.

10       **【Claim 54】** A design verification apparatus comprising:  
a verification software and one or more simulators, wherein said verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one or more  
15 design objects in said design code or synthesized net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast while obtaining the visibility, thereby providing both fast simulation speed and high visibility for debugging.

20       **【Claim 55】** A design verification method, in which one or more design bugs in the design code or synthesized gate-level net-list are identified and collected by a number of simulation executions with a number of test benches, comprising;  
some of said simulation executions is decomposed into the 1<sup>st</sup> simulation run as the front-state simulation and the post-1<sup>st</sup> simulation runs as the back-stage simulation, verification  
25 software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> simulation runs as the back-stage simulation can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-state simulation, and said one or  
30 more post-1<sup>st</sup> simulation runs are executed fast while obtaining the visibility, thereby providing both fast simulation speed and high visibility for debugging.

**【Claim 56】** The design verification method, in which there is no change in the design objects, the minimal simulation results are saved during a run of the front-stage  
35 simulation, the saved simulation results are used for running one or more back-stage simulation either in parallel with two or more simulators or in sequence with a single simulator, and if necessary the dump on one or more variables or signals in one or more design objects is carried out during the back-stage simulation, thereby achieving high visibility.

【Claim 57】 The design verification method according to Claim 37, said simulation result at the higher level of abstraction contains one or more simulation waveform files, and those waveform files are used in simulation at the lower level of abstraction.

5      【Claim 58】 The design verification method according to Claim 37, as said simulation result at the higher level of abstraction one or more simulation waveform files exist, and those waveform files are used in simulation at the lower level of abstraction.

10      【Claim 59】 The design verification and correction method according to Claim 37 wherein assuming one of two simulation results at the higher level of abstraction and the lower level of abstraction as a reference, comparing the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction, finding the differences between the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction and making them correct, thereby said  
15      two simulation results compatible.

20      【Claim 60】 The design verification and correction method according to Claim 44, or Claim 59 wherein comparing the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction at one or more specific simulation times, finding the differences between the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction and making them correct by changing the simulation results of the lower level of abstraction to be compatible with the simulation results of the higher level of abstraction.

25      【Claim 61】 The design verification and correction method according to Claim 44, or Claim 59 wherein comparing the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction at one or more specific simulation times, finding the differences between the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction and making them correct by  
30      changing the simulation results of the higher level of abstraction to be compatible with the simulation results of the lower level of abstraction.

35      【Claim 62】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is transaction-based simulation, and said simulation at the lower level of abstraction is cycle-based simulation.

40      【Claim 63】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is behavioral level simulation, and said simulation at the lower level of abstraction is register transfer level simulation.

5       【Claim 64】 The design verification method according to Claim 37, Claim 57, or Claim 58 wherein said simulation results at the higher level of abstraction contains the design state information of one or more design objects and the information of all inputs and inouts in input mode of said one or more design objects during the entire simulation period.

10       【Claim 65】 The design verification method according to Claim 38, or Claim 64 wherein said design state information of one or more design objects is not one for the entire simulation period, but one at one or more specific simulation times in the simulation period.

15       【Claim 66】 The design verification method according to Claim 37, Claim 57, or Claim 58 wherein said simulation results at the higher level of abstraction contains the information of all inputs and inouts in input mode of said one or more design objects during the entire simulation period.

20       【Claim 67】 The design verification method according to Claim 66 wherein two or more simulation runs at the lower level of abstraction are executed in parallel by using said information of all inputs and inouts in input mode of said one or more design objects during the entire simulation period so that the simulation at the lower level of abstraction are executed fast.

25       【Claim 68】 The design verification method according to Claim 41, or Claim 67 wherein said parallel execution uses both temporally parallel execution and spatially parallel execution.

30       【Claim 69】 The design verification method according to Claim 37, Claim 38, and Claim 64 wherein two or more simulation runs at the lower level of abstraction are executed in parallel by using both said design state information and said information of all inputs and inouts in input mode of said one or more design objects during the entire simulation period so that the simulation at the lower level of abstraction are executed fast.

35       【Claim 70】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is transaction-based simulation, and said simulation at the lower level of abstraction is event-driven simulation.

40       【Claim 71】 The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction uses both transaction-based simulation and cycle-based simulation together, and said simulation at the lower

level of abstraction is event-driven simulation.

5       **【Claim 72】** The design verification method according to Claim 28 wherein at least part of said translated design code, which has different syntax from the original code but is functionally equivalent to the original, is obtained by logic synthesis or logic transformation.

10       **【Claim 73】** The design verification method according to Claim 28 wherein at least part of said translated design code, which has different syntax from the original code but is functionally equivalent to the original, is represented by binary decision diagram, or multi-valued decision diagram, and simulated.

15       **【Claim 74】** The design verification method, in which a simulation at the higher level of abstraction is executed such a way that its simulation result is entirely or partially corrected by using the simulation result at the lower level of abstraction.

20       **【Claim 75】** The design verification and correction method according to Claim 37, or Claim 74 wherein comparing the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction, finding the differences between the simulation result of the lower level of abstraction and the simulation result of the higher level of abstraction and making them correct by conducting one or more partial simulation where they are different, thereby said two simulation results compatible.

25       **【Claim 76】** The design verification method according to Claim 75 wherein in said one or more partial simulation the simulation at the lower level of abstraction uses the simulation result of the higher level of abstraction, the simulation at the higher level of abstraction uses the simulation result of the lower level of abstraction just obtained, and these processes are repeated until said two simulation results compatible.

30       **【Claim 77】** The design verification and correction method according to Claim 44, or Claim 59 wherein said simulation at the higher level of abstraction is gate-level simulation without timing information, and said simulation at the lower level of abstraction is gate-level timing simulation using timing simulation.

35       **【Claim 78】** The design verification method according to Claim 47, or Claim 77 wherein if said gate-level simulation is not timing simulation, then at least part of design objects is represented by binary decision diagram, or multi-valued decision diagram, and simulated.

40       **【Claim 79】** The design verification method according to Claim 76 wherein simulation period is narrowed down as said repeated processes, in which the simulation at the lower

level of abstraction uses the simulation result of the higher level of abstraction, the simulation at the higher level of abstraction uses the simulation result of the lower level of abstraction just obtained, and so on, go on until said two simulation results compatible.

5       **【Claim 80】** A design verification apparatus comprising:  
a verification software and at least two or more different verification platforms, wherein  
said verification software instruments the additional code or circuit to the design code or  
synthesized net-list in an automatic way so that the minimally necessary information, with  
which one or more post-1<sup>st</sup> verification runs as the back-stage verification on the  
10      verification platforms among which at least one verification platform is different from the  
verification platform for the 1<sup>st</sup> verification run can be executed against one or more  
design objects in said design code or synthesized net-list, can be collected during the 1<sup>st</sup>  
verification run, which is the front-stage verification, and said one or more post-1<sup>st</sup>  
verification runs are executed fast.

15       **【Claim 81】** A design verification apparatus according to Claim 80 wherein said  
verification software and said at least two or more different verification platforms, wherein  
said verification software instruments the additional code or circuit to the design code or  
synthesized net-list in an automatic way so that the minimally necessary information, with  
20      which one or more post-1<sup>st</sup> verification runs as the back-stage verification on the  
verification platforms among which at least one verification platform is different from the  
verification platform for the 1<sup>st</sup> verification run can be executed against one or more  
design objects in said design code or synthesized net-list, can be collected during the 1<sup>st</sup>  
verification run, which is the front-stage verification, and said one or more post-1<sup>st</sup>  
25      verification runs can be executed fast as most of those runs can be started at other  
simulation times than the simulation time 0.

30       **【Claim 82】** A design verification apparatus according to Claim 80 wherein said  
verification software instruments the additional code or circuit to the design code or  
synthesized net-list in an automatic way so that the minimally necessary information, with  
which one or more post-1<sup>st</sup> verification runs as the back-stage verification on the  
verification platforms among which at least one verification platform is different from the  
verification platform for the 1<sup>st</sup> verification run can be executed against one or more  
design objects in said design code or synthesized net-list, can be collected during the 1<sup>st</sup>  
35      verification run, which is the front-stage verification, and said one or more post-1<sup>st</sup>  
verification runs can be executed fast as they are only executed against one or more  
specific blocks, not for all blocks.

40       **【Claim 83】** A design verification method, in which one or more design bugs in the  
design code or synthesized gate-level net-list are identified and corrected by running a

number of verification executions using at least two or more verification platforms in hybrid way, comprising;

some of said verification executions is decomposed into the 1<sup>st</sup> verification run as the front-stage verification and the post-1<sup>st</sup> verification runs as the back-stage verification, the additional code or circuit is instrumented to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> verification runs as the back-stage verification on the verification platforms among which at least one verification platform is different from the verification platform for the 1<sup>st</sup> verification run can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast as they are executed against one or more specific blocks only.

**【Claim 84】** A design verification method, in which one or more design bugs in the design code or synthesized gate-level net-list are identified and corrected by running a number of verification executions using at least two or more verification platforms in hybrid way, comprising;

some of said verification executions is decomposed into the 1<sup>st</sup> verification run as the front-stage verification and the post-1<sup>st</sup> verification runs as the back-stage verification, the additional code or circuit is instrumented to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> verification runs as the back-stage verification on the verification platforms among which at least one verification platform is different from the verification platform for the 1<sup>st</sup> verification run can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> simulation run, which is the front-stage simulation, and said one or more post-1<sup>st</sup> simulation runs are executed fast as most of those runs can be started at other simulation times than the simulation time 0.

**【Claim 85】** A design verification method according to Claim 83, or Claim 84 wherein said one or more post-1<sup>st</sup> verification runs as the back-stage verification on the verification platforms among which at least one verification platform is different from the verification platform for the 1<sup>st</sup> verification run are executed in parallel on at least two or more verification platforms.

**【Claim 86】** A design verification method according to Claim 85 wherein said one or more post-1<sup>st</sup> verification runs as the back-stage verification on the verification platforms among which at least one verification platform is different from the verification platform for the 1<sup>st</sup> verification run are executed in parallel on at least two or more verification platforms, and these at least two or more verification platforms are different platforms.

5       **【Claim 87】** A design verification method according to Claim 83, or Claim 84 wherein said one or more post-1<sup>st</sup> verification runs as the back-stage verification on the verification platform which is different from the verification platform for the 1<sup>st</sup> verification run is sequentially executed.

10       **【Claim 88】** The design verification method according to Claim 83, Claim 84, Claim 85, Claim 86, or Claim 87 wherein the post-1<sup>st</sup> verification runs are started with a verification platform from two or more saved design states one by one sequentially which have stored at 1<sup>st</sup> verification run as the front-stage verification, one design state, which is the latest design state in the verification time, is selected and used so that first of many post-1<sup>st</sup> verification runs can start from the latest design state selected and during the verification all signals and variables in the design or synthesized gate-level net-list are probed, and more post-1<sup>st</sup> verification runs are also executed in the reverse order of design state saving times.

15       **【Claim 89】** The design verification method according to Claim 80, Claim 81, Claim 82, Claim 83, or Claim 84 wherein the collection of said minimally necessary information at the 1<sup>st</sup> verification run as the front-stage verification includes to save design states of DUV at some fixed interval, or one or more specific verification times, and the values on all inputs and inouts of DUV in the entire verification run period into file.

20       **【Claim 90】** The design verification method according to Claim 80, Claim 81, Claim 82, Claim 83, or Claim 84 wherein the collection of said minimally necessary information at the 1<sup>st</sup> verification run as the front-stage verification includes to save design states of DUV and the states of TB at some fixed interval, or one or more specific verification times into file.

25       **【Claim 91】** The design verification method according to Claim 80, Claim 81, Claim 82, Claim 83, or Claim 84 wherein the collection of said minimally necessary information at the 1<sup>st</sup> verification run as the front-stage verification includes to save simulation states at some fixed interval, or one or more specific verification times into file if the 1<sup>st</sup> verification run is carried out with at least one or more simulators.

30       **【Claim 92】** The design verification method according to Claim 91 wherein said one or more saved simulation states as the collection of said minimally necessary information at the 1<sup>st</sup> simulation run as the front-stage verification is done by using the save command of simulators, and the post-1<sup>st</sup> verification runs start from the design states which are extract from said one or more saved simulation states.



5       **【Claim 93】** The design verification method according to Claim 80, Claim 81, Claim 82, Claim 83, or Claim 84 wherein the collection of said minimally necessary information at the 1<sup>st</sup> verification run as the front-stage verification includes to prove and save the values on all inputs and inouts of specific design blocks existed in the design code or synthesized design net-list during the entire verification run period into file.

10       **【Claim 94】** The design verification method according to Claim 85, or Claim 86 wherein each of one or more post-1<sup>st</sup> verification runs, which is the back-stage verification, is transformed into two or more verification sub-runs which use two or more DUV design states saved during the 1<sup>st</sup> verification run, which is the front-stage verification, and said two or more verification sub-runs are independently running in parallel on two or more verification platforms which are installed on two or more computers connected into a computer network, and merging the verification results on said two or more verification platforms together as the entire verification result, thereby achieving a fast verification.

15       **【Claim 95】** The design verification method according to Claim 85, or Claim 86 wherein each of one or more post-1<sup>st</sup> verification runs, which is the back-stage verification, is transformed into two or more verification sub-runs which use two or more simulation executable files compiled from the files of design blocks in DUV and their input and input values probed and saved during the 1<sup>st</sup> verification run, which is the front-stage verification, and said two or more verification sub-runs are independently running in parallel on two or more verification platforms which are installed on two or more computers connected into a computer network, and merging the verification results on said two or more verification platforms together as the entire verification result, thereby achieving a fast verification.

20       **【Claim 96】** A design verification apparatus according to Claim 82, wherein said one or more post-1<sup>st</sup> verification runs as the back-stage verification on the verification platforms which is different from the verification platform for the 1<sup>st</sup> verification run can be executed against one or more design objects in said design code or synthesized net-list, can be collected during the 1<sup>st</sup> verification run, which is the front-stage verification, and said one or more post-1<sup>st</sup> verification runs can be executed fast as they are only executed against one or more specific blocks, not for all blocks.

30       **【Claim 97】** The design verification method according to Claim 83 wherein the collection of said minimally necessary information at the 1<sup>st</sup> verification run as the front-stage verification includes to probe on all inputs and inouts in one or more design blocks existed in the design code or synthesized gate-level net-list during the entire verification and save them into files, they are compiled with one or more design blocks existed in the design code or synthesized gate-level net-list in a verification platform for generating one

40

or more verification execution files, and one of them is executed while additional probing is applied to said one or more design blocks existed in the design code or synthesized gate-level net-list at the post-1<sup>st</sup> verification runs which are the back-stage verification, thereby providing both fast verification and visibility inside the design blocks.

5

【Claim 98】 A design verification method, in which the additional verification uses the results from previous execution of arbitrary simulation, simulation acceleration, hardware emulation, or prototyping comprising, and said additional verification is decomposed into the 1<sup>st</sup> verification run as the front-stage verification and the post-1<sup>st</sup> verification runs as the back-stage verification, verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the minimally necessary information, with which one or more post-1<sup>st</sup> verification runs as the back-stage verification can be executed against one or more design objects in said design code or synthesized gate-level net-list, can be collected during the 1<sup>st</sup> verification run, which is the front-stage verification, and said one or more post-1<sup>st</sup> verification runs are executed fast.

10

15

25

【Claim 99】 A design verification apparatus comprising:  
a verification software and at least one or more verification platforms, wherein said verification software instruments the additional code or circuit to the design code or synthesized net-list in an automatic way so that the dynamic information can be collected during one or more verification runs, and said dynamic information collected is re-used at the post-debugging simulation after at least one design object is changed for debugging, thereby total verification time can be entirely or partially reduced.

20

30

【Claim 100】 A design verification method comprising:  
by using a verification software and at least one or more verification platforms the additional code or circuit is instrumented to the design code or synthesized net-list in an automatic way so that the dynamic information can be collected during one or more verification runs, and said dynamic information collected is re-used at the post-debugging simulation after at least one design object is changed for debugging, thereby total verification time can be entirely or partially reduced.

35

【Claim 101】 A design verification method according to Claim 99, or Claim 100 wherein the detection method of finding verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change is to compare all values of outputs and inouts of said design object before and after the change in automatic way.

40

【Claim 102】 A design verification method according to Claim 99, or Claim 100 wherein

the detection method of finding verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change is to compare all values of outputs of said design object before and after the change in automatic way.

5

【Claim 103】 A design verification method according to Claim 99, or Claim 100 wherein the detection method of finding verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change is to apply the re-simulation input stimuli, which is obtained in one or more verification runs before modification, to said changed design object in automatic way.

10

【Claim 104】 A design verification method according to Claim 99, Claim 100, Claim 101, Claim 102, or Claim 103 wherein the verification run after at least one design objects is modified is executed only with said at least one design object changed for debugging and its re-execution input stimuli at least to the first verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change from the verification time 0.

15

【Claim 105】 A design verification method according to Claim 99, Claim 100, Claim 101, Claim 102, or Claim 103 wherein the verification run after at least one design objects is modified is executed only with said at least one design object changed for debugging, its re-execution input stimuli, and the part of design objects unchanged at least to the first verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change from the verification time 0.

20

25

【Claim 106】 A design verification method according to Claim 99, Claim 100, Claim 101, Claim 102, or Claim 103 wherein the verification run after at least one design objects is modified is executed with all design objects after first verification time of different results when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change in automatic way.

30

【Claim 107】 A design verification method according to Claim 99, Claim 100, Claim 101, Claim 102, or Claim 103 wherein the verification run after at least one design objects is modified is executed with all design objects from the verification time of different results when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change to the verification time of different results in automatic way.

35

40

5       【Claim 108】 A design verification method according to Claim 99, Claim 100, Claim 101, Claim 102, or Claim 103 wherein the verification run after at least one design objects is modified is executed only with said at least one design object changed for debugging and its re-execution input stimuli at least to the first verification time of different results when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change from the verification time 0, and after the first verification time of different results, and the verification run is automatically switched to the verification execution with all  
10       design.

15       【Claim 109】 A design verification method according to Claim 106, or Claim 108 wherein to automatically switch the verification run with all of design objects from said first verification time for different results, the restoring design states for the design objects unchanged is occurred with the design state information saved during the verification runs before the modification for debugging at the boundary of said first verification time for different results.

20       【Claim 110】 A design verification method according to Claim 109 wherein said restoring design objects is by re-start capability of simulators, and design states saving is by snapshot capability of simulators.

25       【Claim 111】 A design verification method according to Claim 110 wherein for said re-start capability of simulators the restore or restart command is used, and for said snapshot capability the save or checkpoint command is used.

30       【Claim 112】 A design verification method according to Claim 99, or Claim 100 wherein the detection method of finding verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change is to sequentially compare all values of outputs and inouts of said design object before and after the change in automatic way.

35       【Claim 113】 A design verification method according to Claim 99, or Claim 100 wherein the detection method of finding verification time when the verification result of at least one design object changed for debugging is not same as the verification result of the said design object before the change is to compare all values of outputs and inouts of said design object in parallel before and after the change in automatic way.

40       【Claim 114】 A design verification apparatus according to Claim 99 wherein said verification platforms are simulators, simulation accelerators, emulators, prototyping

systems, or their combination.

5       **【Claim 115】** A design verification apparatus according to Claim 99, Claim 100, Claim 101, Claim 102, Claim 103, Claim 104, Claim 105, Claim 110, or Claim 111 wherein if said verification run are done entirely or partially, then two or more processes are co-executed by inter-process communication for an entire verification run.

10       **【Claim 116】** A design verification method according to Claim 115 wherein at least one process among said one or more processes which are co-executed by inter-process communication collects dynamic information of one or more design objects during the verification runs before the design code modification by using snapshot, and the verification runs at the specific verification times after the design code modification are carried out by re-startability using said dynamic information collected, thereby reducing the verification time entirely or partially.

15       **【Claim 117】** A design verification method according to Claim 116 wherein for said snapshot the save or snapshot capability in simulators is used, and for said re-startability the restart or restore capability in simulators is used.

20       **【Claim 118】** A design verification method according to Claim 99, Claim 100, Claim 101, Claim 102, Claim 103, Claim 104, Claim 105, Claim 106, Claim 108, Claim 109, Claim 115, Claim 116, or Claim 117 wherein the verification platform contains either the simulation accelerator or the verification platforms containing the simulation accelerator.

25       **【Claim 119】** A design verification method according to Claim 106, Claim 108, Claim 109, Claim 110, or Claim 111 wherein said switching time during the verification run after design code modification is determined by the instrumented code, which is added to the design object, during the verification run in automatic way.

30       **【Claim 120】** A design verification method according to Claim 106, Claim 108, Claim 109, Claim 110, or Claim 111 wherein said alignment of the dynamic information of design objects unmodified and modified at said switching time during the verification run after design code modification is determined by the instrumented code, which is added to the design object, during the verification run in automatic way.

35       **【Claim 121】** A design verification method according to Claim 106, Claim 108, Claim 109, Claim 110, or Claim 111 wherein said verification run with all design objects after said switching time after design code modification is determined by the instrumented code, which is added to the design object, during the verification run in automatic way.

40